National University of Singapore
School of Computing

IS2103 Enterprise Systems Development Concepts

Part 1: RMI and RMI-IIOP
Module Search

Total Marks: 10

Part 1a (RMI): 4 marks

Implement the `ModuleSearch` client and server applications as described in pages 2 to 5 of this document.

Part 1b (RMI-IIOP): 6 marks

Implement the `ModuleSearch` client and server applications as described in pages 6 to 9 of this document.

Submission of Part 1a and 1b

The folders for Part 1a and 1b should include all ".java" and ".class" files and any data files that you may have for the parts. Instructions on *how* to submit the zip file and *how* the assignment will be evaluated are available at the IS2103 IVLE page under the Assessment heading.

Evaluation of Part 1a and 1b

You will be evaluated on the implementation of the clients and servers of the two applications. You should therefore test them thoroughly to ensure they work in accordance with the specification of the two applications documented below.

Part 1a: Remote Method Invocation (RMI)

Module Search

A `ModuleSearchServer` exposes the following interface to its clients:

```
package modulesearch;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.ArrayList;

public interface ModuleSearchServer extends Remote {

    //input : module code (String)
    //output: title of this module (String)
    public String getTitle(String moduleCode)
            throws RemoteException;

    //input : module title (String)
    //output: lecturer's name of this module (String)
    public String getLecturer(String moduleTitle)
            throws RemoteException;

    //input : module level (int)
    //output: list of module codes in this level (ArrayList<String>)
    public ArrayList<String> getModuleList(int moduleLevel)
            throws RemoteException;

}
```

You are required to develop an application that can be used for getting module information in certain level, for example, get module codes, module titles and lecturer names of all 4000 modules.

The interface defines three methods:

1. The `getTitle()` method returns the module title of the given module code. The matching of module codes and module titles is stored in "`module.txt`" file (See Appendix A). If the module code is not found in the server, a "NULL" value is returned.
2. The `getLecturer()` method returns the lecturer's name of the given module title. The matching of module titles and lecturers is stored in "`lecturer.txt`" file (See Appendix A). If the module title is not found in the server, a "NULL" value is returned.
3. The `getModuleList()` method returns a list of all module codes in a given level. The value of module level can only be: 1000, 2000, 3000, 4000, 5000 and 6000. If the module level is not found in the server, a "NULL" value is returned.

You are required to implement the `ModuleSearchServer` interface. You may implement the methods in whatever way you deem fit subject to the following conditions:

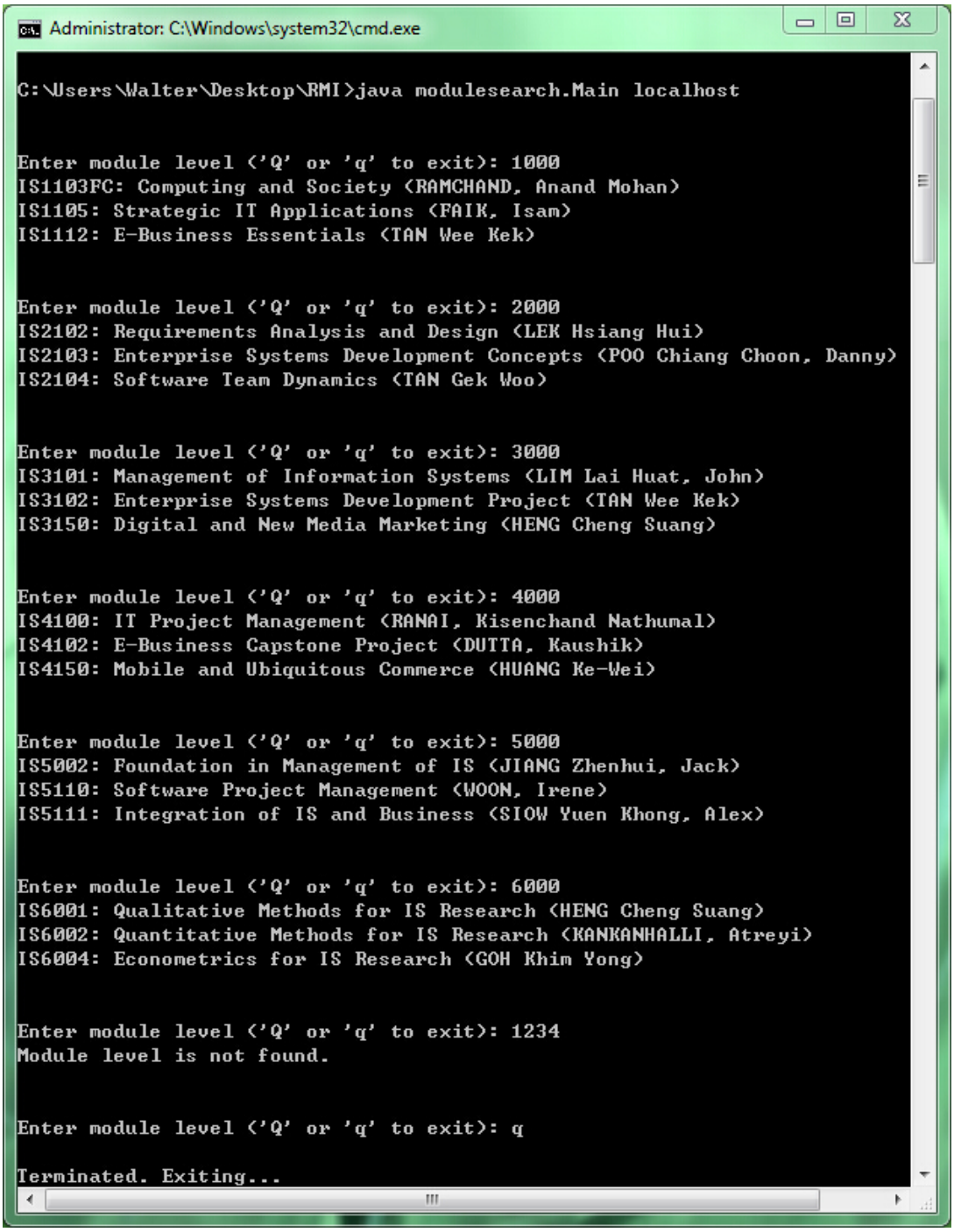
1. There are 18 modules belonging to 6 levels (from 1000 to 6000) provided. You are required to use "`module.txt`", "`lecturer.txt`" in implementing the server.
2. Name the implementation class of the `ModuleSearchServer` interface as

`ModuleSearchServerImpl`. Produce an accompanying server application class to start the server. Name the server application as `ModuleSearchServerApp`.

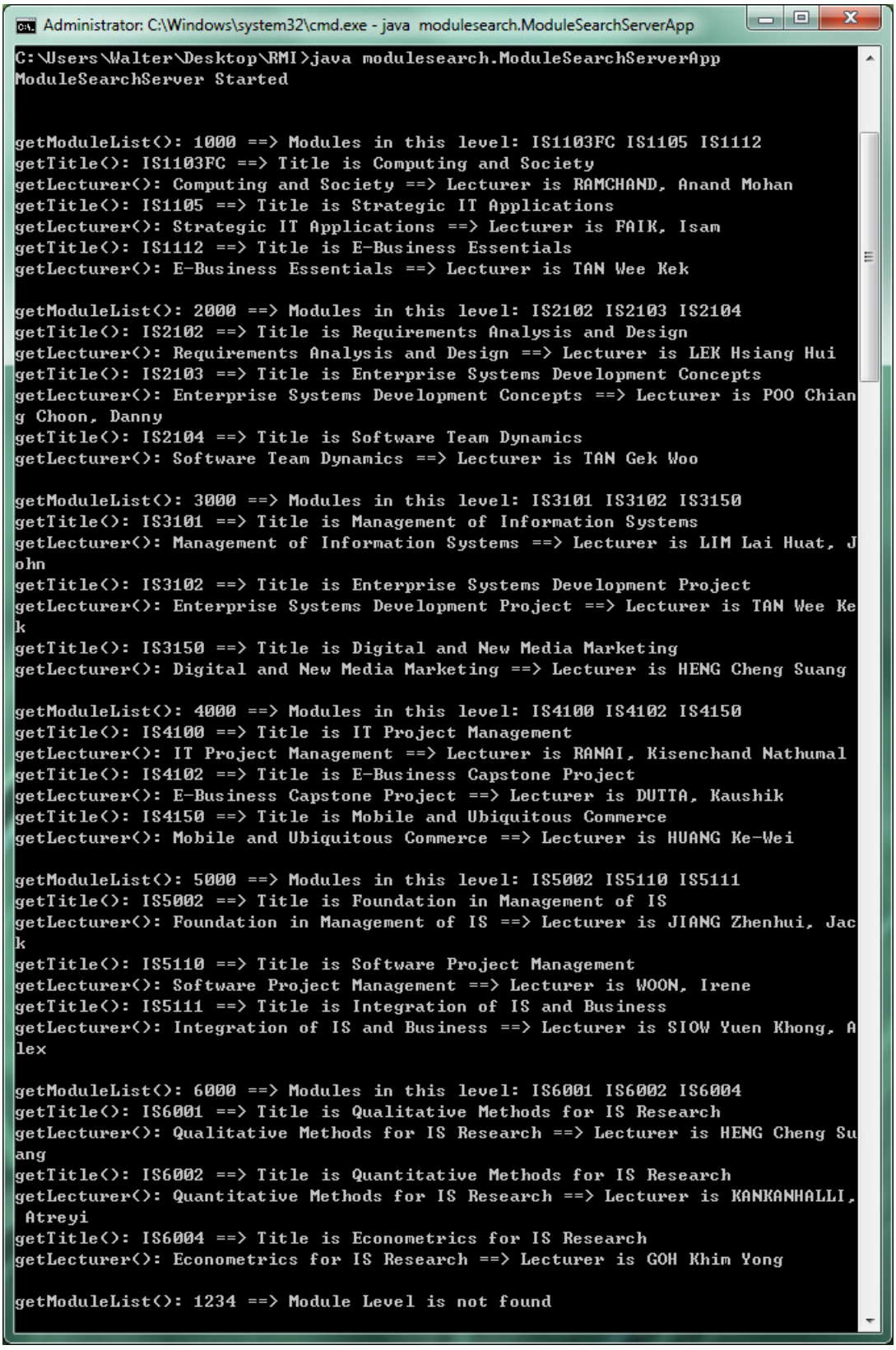
Create a client application and name it as `Main`. Your client application should be able to demonstrate the following functionalities:

- Able to connect to the server via the RMI protocol using the default port 1099. For Macintosh users, you may use other port if necessary.
- It has one argument "`localhost`" entered at the DOS prompt. This argument denotes the server's IP address.
- Able to repeatedly call the server's `getTitle()` method to enquire on the module title of a module code, `getLecturer()` method to get the lecturer's name of a given module title, `getModuleList()` method to get a list of module codes in a given level. If modules are found in the given level, a list of module details, including module codes, module titles and lecturer names, will be displayed. If an invalid module level is requested, the client application will display "Module level is not found".
- Able to terminate the repeated call when the user is done with your client application.

The following is a series of outputs at the client side:



The following is the corresponding outputs at the server side:



Part 1b: RMI-IIOP

Module Search

A `ModuleSearchServer` exposes the following interface to its clients:

```
package modulesearch;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.ArrayList;

public interface ModuleSearchServer extends Remote {

    //input : module code (String)
    //output: title of this module (String)
    public String getTitle(String moduleCode)
            throws RemoteException;

    //input : module title (String)
    //output: lecturer's name of this module (String)
    public String getLecturer(String moduleTitle)
            throws RemoteException;

    //input : module level (int)
    //output: list of module codes in this level (ArrayList<String>)
    public ArrayList<String> getModuleList(int moduleLevel)
            throws RemoteException;

}
```

You are required to develop an application that can be used for getting module information in certain level, e.g. get module codes, module titles and lecturer names of all 4000 modules.

The interface defines three methods:

4. The `getTitle()` method returns the module title of the given module code. The matching of module codes and module titles is stored in "`module.txt`" file (See Appendix A). If the module code is not found in the server, a "NULL" value is returned.
5. The `getLecturer()` method returns the lecturer's name of the given module title. The matching of module titles and lecturer is stored in "`lecturer.txt`" file (See Appendix A). If the module title is not found in the server, a "NULL" value is returned.
6. The `getModuleList()` method returns a list of all module codes in a given level. The value of module level can only be: 1000, 2000, 3000, 4000, 5000 and 6000. If the module level is not found in the server, a "NULL" value is returned.

You are required to implement the `ModuleSearchServer` interface. You may implement the methods in whatever way you deem fit subject to the following conditions:

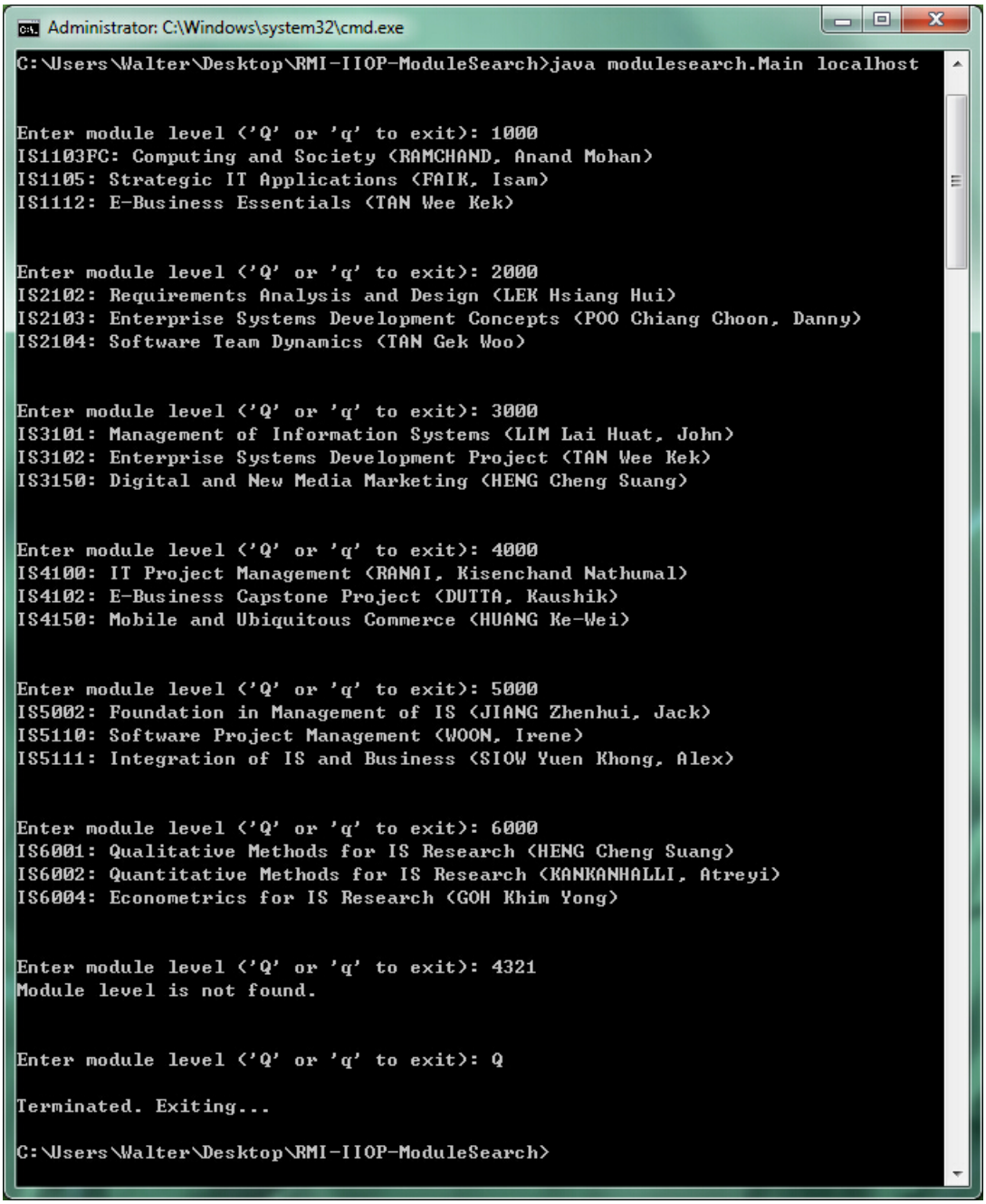
3. There are 18 modules belonging to 6 levels (from 1000 to 6000) listed. You are required to use "`module.txt`", "`lecturer.txt`" in implementing the server.
4. Name the implementation class of the `ModuleSearchServer` interface as

`ModuleSearchServerImpl`. Produce an accompanying server application class to start the server. Name the server application as `ModuleSearchServerApp`.

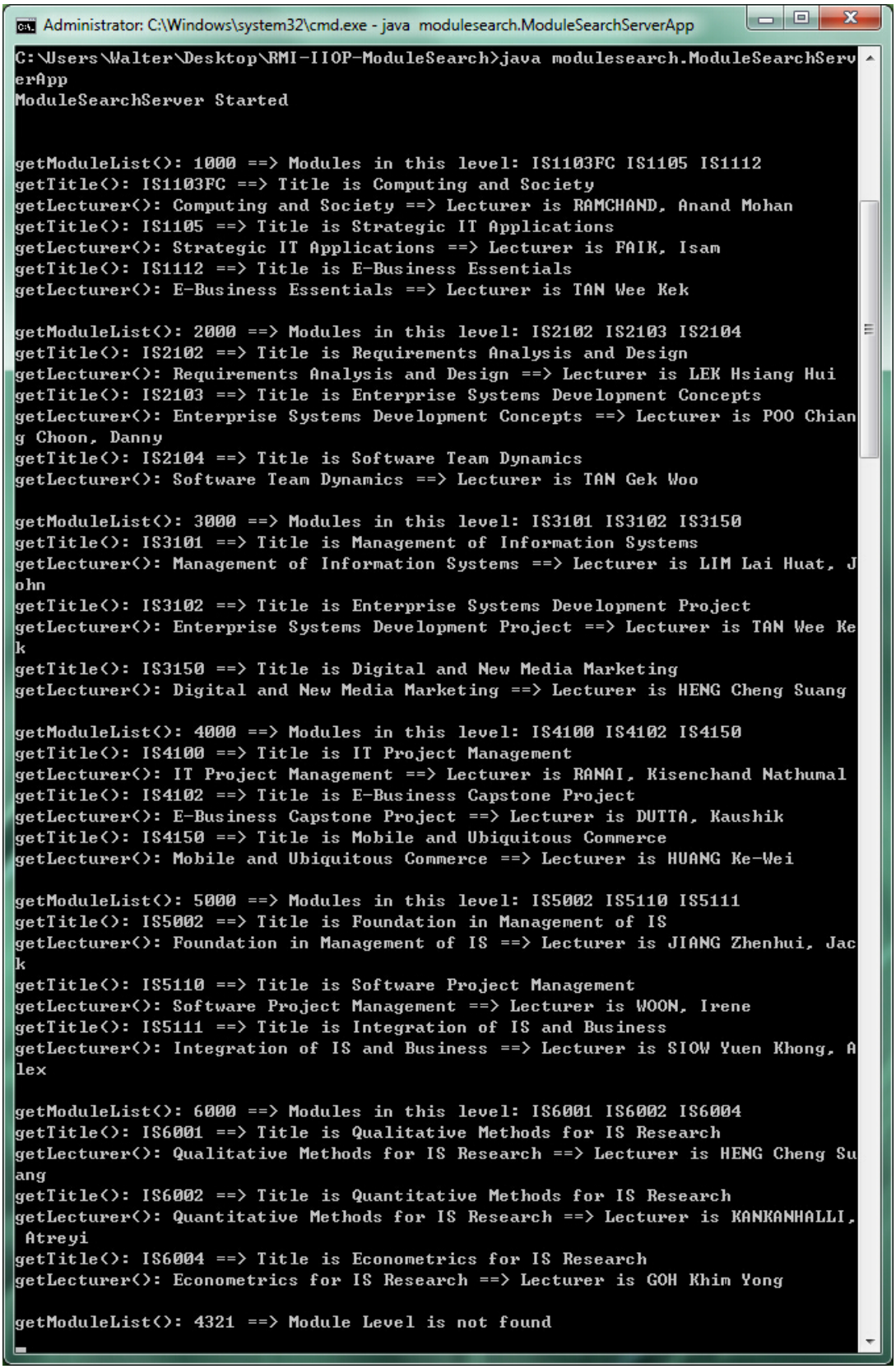
Create a client application and name it as Main. Your client should be able to demonstrate the following functionalities:

- Able to connect to the server via the RMI-IIOP protocol using the default port 900. For Macintosh users, you may use other port if necessary.
- It has one argument "`localhost`" entered at the DOS prompt. This argument denotes the server's IP address.
- Able to repeatedly call the server's `getTitle()` method to enquire on the module title of a module code, `getLecturer()` method to get the lecturer's name of a given module title, `getModuleList()` method to get a list of module codes in a given level. If modules are found in the given level, a list of module details, including module codes, module titles and lecturer names, will be displayed. If an invalid module level is requested, the client application will display "Module level is not found".
- Able to terminate the repeated call when the user is done with your client application.

The following is a series of outputs at the client side:



The following is the corresponding outputs at the server side:



Appendix A: Code-Title List, Title-Lecturer List

Content of "module.txt" (module_code:module_title):

IS1103FC:Computing and Society
IS1105:Strategic IT Applications
IS1112:E-Business Essentials
IS2102:Requirements Analysis and Design
IS2103:Enterprise Systems Development Concepts
IS2104:Software Team Dynamics
IS3101:Management of Information Systems
IS3102:Enterprise Systems Development Project
IS3150:Digital and New Media Marketing
IS4100:IT Project Management
IS4102:E-Business Capstone Project
IS4150:Mobile and Ubiquitous Commerce
IS5002:Foundation in Management of IS
IS5110:Software Project Management
IS5111:Integration of IS and Business
IS6001:Qualitative Methods for IS Research
IS6002:Quantitative Methods for IS Research
IS6004:Econometrics for IS Research

Content of "lecturer.txt" (module_title:lecturer_name):

Computing and Society:RAMCHAND, Anand Mohan
Strategic IT Applications:FAIK, Isam
E-Business Essentials:TAN Wee Kek
Requirements Analysis and Design:LEK Hsiang Hui
Enterprise Systems Development Concepts:POO Chiang Choon, Danny
Software Team Dynamics:TAN Gek Woo
Management of Information Systems:LIM Lai Huat, John
Enterprise Systems Development Project:TAN Wee Kek
Digital and New Media Marketing:HENG Cheng Suang
IT Project Management:RANAI, Kisenchand Nathumal
E-Business Capstone Project:DUTTA, Kaushik
Mobile and Ubiquitous Commerce:HUANG Ke-Wei
Foundation in Management of IS:JIANG Zhenhui, Jack
Software Project Management:WOON, Irene
Integration of IS and Business:SIOW Yuen Khong, Alex
Qualitative Methods for IS Research:HENG Cheng Suang
Quantitative Methods for IS Research:KANKANHALLI, Atreyi
Econometrics for IS Research:GOH Khim Yong